Creating Energy-Efficient Software

by Bob Steigerwald, Rajshree Chabukswar, Karthik Krishnan, Jun De Vega

Table of Contents

Abstract

This paper examines software methodologies, designs, and software development tools that can be used to improve the energy efficiency of application software and extend mobile platform battery time. Computational efficiency, data efficiency, and context-aware methods can all contribute to creating applications that are power-aware. There are many additional resources available in the form of white-papers, developer kits, and analysis tools. These are referenced in the paper and in the References section.

Introduction

For years mobile platform vendors have sought means to extend the battery life for mobile platforms. Battery technologies have gradually improved, processors have new low-power states, and displays have dramatically improved their power consumption. There is still room for improvement. Software can play an important role in reducing the power used on mobile platforms and extend the battery time.

The purpose of this paper is to explain the software methodologies and designs that can be used today to save energy and extend mobile platform battery time as well as describe various tools that support the development of energy-efficient software.

The methodologies described here have been researched and tested by Intel software Application Engineers. In each case, we document the results of the experiments and provide a reference to more detailed information. For a look into the typical test environment and the mechanisms for measuring total platform power, see Appendix A.

The remainder of the paper covers the following topics:

- Background – some fundamentals about power, energy, and the platform power profile
- Computational Efficiency – methods to reduce energy costs by improving application performance
- Data Efficiency – methods reduce energy costs by minimizing data movement and using the memory hierarchy effectively
- Context Awareness – enabling applications to make intelligent decisions at runtime based on the current state of the platform
- Operating Systems – how to take advantage of the resources offered by the OS to save energy
- Tools and Technologies – support for creating energy-efficient applications

Background

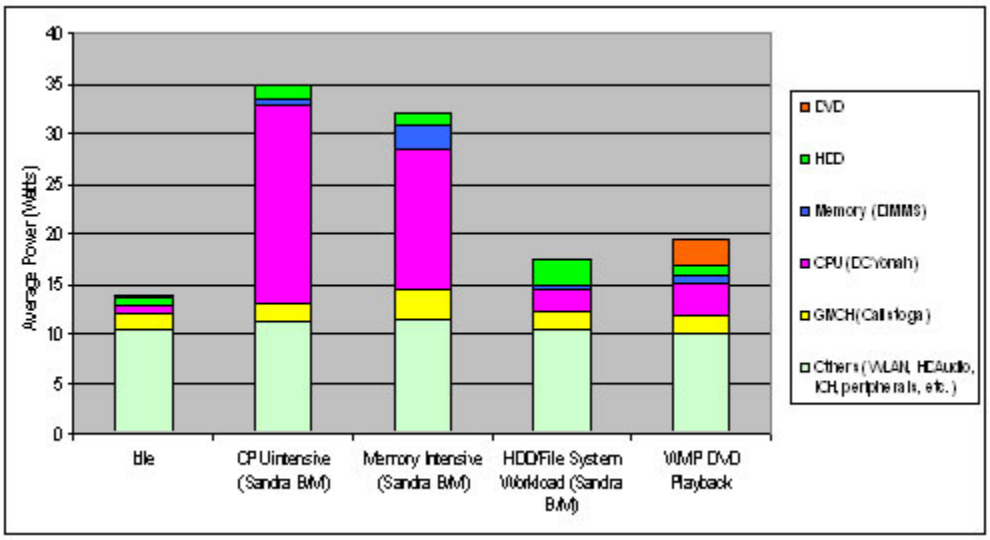
Definitions

Joule – the international standard unit of energy measurement
Energy – The conventional definition of energy is the “capacity to do work”. A device that is energy-efficient requires less energy for its “work” or task than its energy-inefficient counterpart. For this paper, we use the term to mean the amount of joules required to carry out a specific task. For example the energy required to lift a 100 gram object 1 meter against the pull of earth’s gravity is about 1 Joule.
Power – the amount energy consumed per unit of time, typically measured in Watts, where a Watt equals 1 Joule per second. For example, a light bulb rated at 60 Watts consumes 60 Joules in one second. Notebook computers running at their highest energy state are rated between 40 to 60 Watts, but on average consume far less.
Heat – more specifically resistive heat, is a natural by-product of running current through a conductor; engineers strive to minimize this in computer design; too much heat means more cooling (typically by a fan) which requires more energy

While we strive to use the terms energy and power appropriately, there may be instances where they are used synonymously.

Platform Power Profile

The power profile of various components on the mobile platform depends on the usage model. For example, the relative contribution of processor power to the overall platform power will be significant in a CPU-intensive workload, but it will not be a dominant factor while the platform is idling. Furthermore, it may also vary depending on whether both the cores were utilized or not (i.e. single-threaded vs. multithreaded). The following provides an idea of how the profile varies during various usage models. The CPU, memory, and file system tests were run using SiSandra benchmarks (http://www.sisoftware.co.uk). Note that the platform power below does not include LCD, since we have excluded it from our analyses. (Others include WLAN, HD-Audio, mini-card, ICH, and other peripherals.)



As seen above, mobile developers need to have an idea of power drainage depending on the usage model, and target specific components for extending battery life and conserving power.

Computational Efficiency

The goal of computational efficiency is to complete a task more quickly. Intuition tells us that if the CPU can accomplish the task in fewer instructions or by doing work in parallel in multiple cores, and then drop the CPU to a low-power state, then the overall energy required to complete the task will be lower.[1] One approach to achieve this is to use the best algorithms and data structures for the particular problem. Another method, for which we have research results below is to take advantage of the performance per watt advantages of Intel multi-core processors and take full advantage of multi-threading to increase application performance and save energy.

Algorithms

Algorithms and data structures are a long-standing area of research in computer science. Considerable effort has gone into research to find more efficient means to solve problems and to investigate and document the corresponding time and space tradeoffs. While optimizing specific algorithms is not an area of interest for our team per se, we can conclude from computer science theory that the choice of algorithms and data structures can make a vast difference in the performance of an application. All other

[1] It is interesting to note that this is not always true. Due to the quadratic relationship between processor states and voltage, it can be demonstrated that a process running for a longer time at a lower P-state may actually use less total energy than running the same process at a high P-state for less time. This is an area of future research.

things being equal, using an algorithm that computes a solution in O(*n log n*) time is going to perform better than one that does the job in O(n^2) time. For a particular problem, a stack may be better than a queue and a B-tree may be better than a binary tree or a hash function. The best algorithm or data structure to use depends on many factors, which indicates that a study of the problem and a careful consideration of the architecture, design, algorithms, and data structures can lead to an application that performs better and consumes less energy. For a detailed study of the analysis of algorithms see [7,8,9].

Multi-Threading[2]

For mobile platforms, power consumption has always been one of the major areas of importance. With multithreaded applications, the job at hand may be able to finish faster than single-threaded applications. As a result, the boost in performance may result in power savings as system resources will be used for less time, as compared to a single-threaded version. [6]

There are other considerations introduced with multithreading an application, such as the effects on power/performance when the threads in the application are imbalanced (when one thread does significantly more work than the other threads), differences in CPU utilization of the threads (for example, one thread might consume 100 percent CPU, while the other threads might consume 10–20 percent of the CPU), and when the threads are affinitized to a single core rather than running on separate cores. This research investigated such issues with a wide variety of multithreaded applications and multitasking scenarios, and proposes recommendations that should be considered when multithreading an application.

All the tests described here were conducted on dual-core Intel Core Duo engineering sample systems with the (code-name) Napa platform. Power measurements were accomplished with Fluke NetDAQ®[3]

Testing Methodology

A variety of applications (single-threaded and multithreaded implementations), along with test kernels developed in-house, are characterized here for power/performance measurements. These applications include a variety of content creation applications, kernels from the gaming space, kernels using Intel® Integrated Performance Primitives (IPP), and office productivity applications.

Single-threaded and multithreaded implementations of the applications discussed here were tested with two power schemes that Microsoft Windows* XP provides when coupled with Intel SpeedStep® technology: MaxPerf or Always-On (AO) mode and Adaptive or Portable-Laptop (PL) mode. AO mode provides maximum available frequency while PL mode adjusts the frequency to conserve energy.

In the results reported below, the following threading models were tested:

- Data Domain Decomposition: the available data set is divided into separate parts, and each thread works on its individual portion.
- Functional Domain Decomposition: each thread works on separate functionality pieces/sections of code within an application.
- Balanced Threading: each thread has an equal amount of work as other active threads of the application.
- Imbalanced Threading: there is a significant difference in the amount of work done by each thread within an application.

Test Results

The graphs in this section discuss power/performance results on an Intel Core Duo engineering sample system running Windows* XP. Time is expressed in seconds. Power measurements were done with

[2] More detailed coverage of this topic can be found at: http://www.intel.com/technology/magazine/computing/mobile-power-saving-0506.pdf

[3] http://us.fluke.com/usen/products/NetDAQ.htm?catalog_name=FlukeUnitedStates

Fluke NetDAQ, which reports average power (in watts [W]) which is then converted to total power using application run-time data (mWHr).

Balanced Threading

The graph in Figure 1 indicates performance data for running single-threaded (ST) and multi-threaded (MT) versions of several CPU-intensive applications. Cryptography and video encoding applications have two MT implementations and results are indicated as MT-1 and MT-2. For content creation applications, multithreading is done with only one implementation, indicated as MT-1. The multithreaded applications clearly show significant performance improvements over running single-threaded versions. For example, the ST version of cryptography takes ~50 seconds to complete, while both the MT-1 and MT-2 versions take only ~25 seconds.

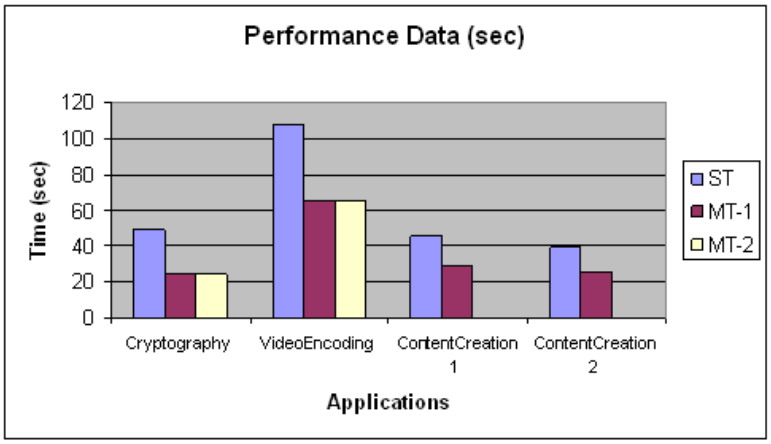


Figure 1: Balanced Threading Performance

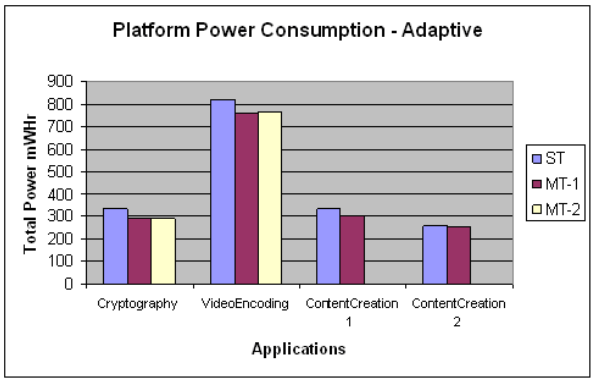


Figure 2: Balanced Threading - CPU Power (Adaptive**)**

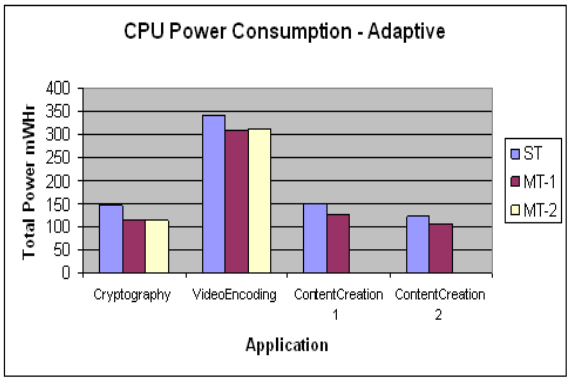


Figure 3: Balanced Threading - Platform Power (Adaptive)

Figures 2 and 3 indicate CPU power and Total Platform power for adaptive (portable/laptop) mode, respectively. Adaptive mode is chosen as it favors power consumption by dynamically changing CPU frequency on demand. For each application run (ST and MT), power data-gathering is normalized to the longest run-time. For example, as indicated in Figure 1, the cryptography workload runs for ~50 seconds in ST mode and ~25 sec in MT modes. The power data is measured for 50 seconds in both ST and MT cases. The intent here is to determine if power can be saved by finishing the CPU-intensive task faster (as in the case of MT) and going to idle state for the remaining duration.

As indicated in Figure 2, power saving is achieved by finishing the job faster (MT) and idling for the remainder of the time as the ST version. This indicates that multithreading done correctly not only shows performance improvements but also saves power. For example, the cryptography ST version running for

~50 seconds consumes ~150 mWHr of total power, while running the cryptography MT version for ~25 seconds and idling the system for the remaining 25 seconds consumes ~110 mWHr of total power. Hence, multithreading helps save power. The graph in Figure 3 indicates the implication of multithreading on total platform power. As indicated, running a multithreaded version of an application consumes lower platform power compared to running a single-threaded version.

Imbalanced Threading

In this section, we will examine power/performance implications on an application with an imbalanced threading model. For this study, a sample game physics engine was created (using Microsoft DirectX*). The sample application has two parts: 1) Physics Computation (collision detection and resolution for graphics objects) and 2) Rendering (updated positions are drawn onto screen). The design of the application was deliberate so that balanced and imbalanced threading could be studied for a CMP (chip multi-processing) processor. Briefly:

- **Balanced**: For this implementation, graphical objects (and background imagery) were divided into two parts, and each thread takes care of the collision detection and resolution of its own set of objects.
- **Imbalanced**: In this implementation, one thread was tasked with performing collision detection and resolution for the colliding objects, while the other thread calculated the updated positions. The result was the desired goal of the first thread being more CPU-intensive than the second thread.

The intent behind creating two multithreaded implementations here is to evaluate power/performance impact when an imbalanced threading model is used, as compared to a balanced threading model. With the two implementations, performance data in different power schemes (MaxPerf, Adaptive, and Adaptive with GV3 fix[4]) are shown in Figure 4.

As indicated on the first two sets in the graph above (Figure 4), performance of imbalanced multithreaded (Imbalanced-MT) implementation degrades from ~64 seconds in MaxPerf mode to ~120 seconds in Adaptive model. However, with the GV3 fix from Microsoft, performance of the Imbalanced-MT completes in less time than ST, as it should. In all cases, Balanced-MT has better performance than Imabalanced-MT.

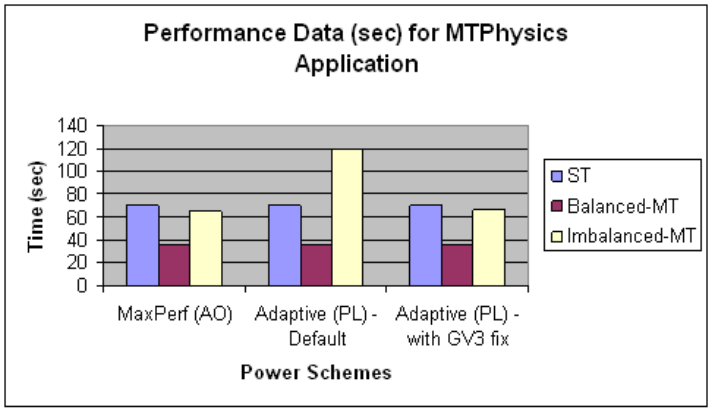


Figure 4: Imbalanced Threading Performance

[4] GV3is a Microsoft hotfix (KB896256) to change the kernel power manager to track CPU utilization across the entire package instead of individual cores. It resolves an issue the power manager had with incorrectly calculate the optimal target performance state for the processor when one core was much less busy than the others. The performance state was set too low and performance suffered in adaptive mode.

Figure 5 shows the results for the measurements of platform power consumption. The power measurements discussed here were normalized using a technique mentioned in the Balanced Threading Model section. As expected, platform power consumption increases with a reduction in the performance in Adaptive (PL)—Default mode. Since the Imbalanced-MT workload now takes much longer to finish, the performance degradation causes an increase in power consumption. With the GV3 fix in place, the improvements in performance yield corresponding improvement in total platform power consumption.

The third set in Figure 4 indicates data with a kernel hotfix. In this case, imbalanced-MT implementation in Adaptive (PL) mode shows similar power/performance data as that of MaxPerf (AO) mode. With this fix, processors run at optimum frequency, not causing degradation in Adaptive (PL) mode. Platform power data with the fix is shown in Figure 5 in the Adaptive (PL)—with GV3 Fix column.

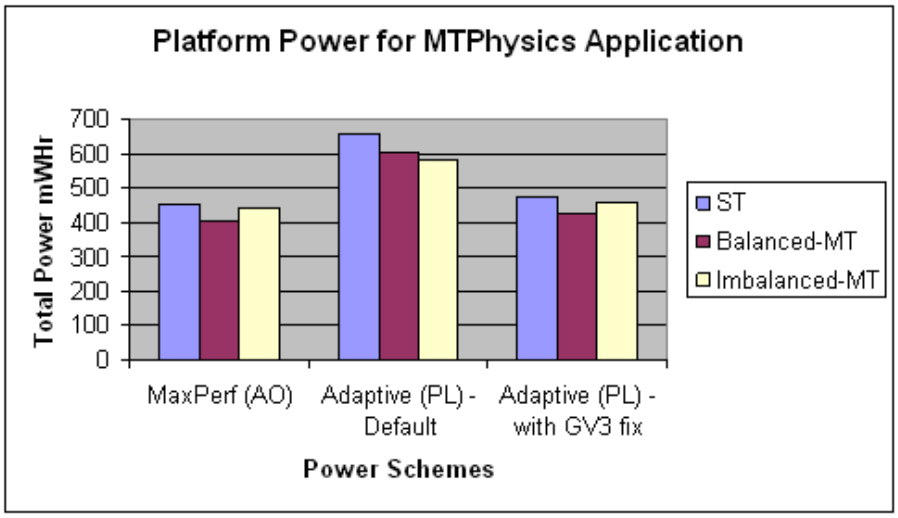


Figure 5: Imbalanced Threading - Platform Power

These results indicate that the imbalanced threading model/under-utilized CPU may cause degradation in performance, causing increased power consumption. We recommend use of a balanced threading model while multithreaded applications. Thread imbalance can be identified by using tools like the Microsoft Perfmon*, the timeline view offered by Intel® VTune™ and Intel® Threading Tools (such as Intel® Thread Checker and Intel® Thread Profiler) to track individual thread run-time and processor utilization counters.

Multitasking Scenarios with one Application Affinitized to Single Core

One of the common usage scenarios for PC users is running multiple applications simultaneously—multitasking. To understand the performance and power impact of running two applications simultaneously, an experiment was conducted with two office productivity applications running concurrently. Since scheduling the two applications using different techniques is likely to show power/performance impacts, the following scenarios were examined:

- Microsoft Windows XP scheduling both applications using its scheduling algorithm (no affinitization).
- Each application hard-affinitized to each core. Application 1 runs on core 0 and application 2 runs on core 1.
- One of the applications hard-affinitized to Core 0 while Windows XP schedules the other application.

These scheduling configurations were chosen to identify if a certain scheduling mechanism favors both power and performance as compared to the others.

Performance data for these configurations is shown in Figure 6. As indicated in the graph, there is no significant performance difference observed with different scheduling configurations.

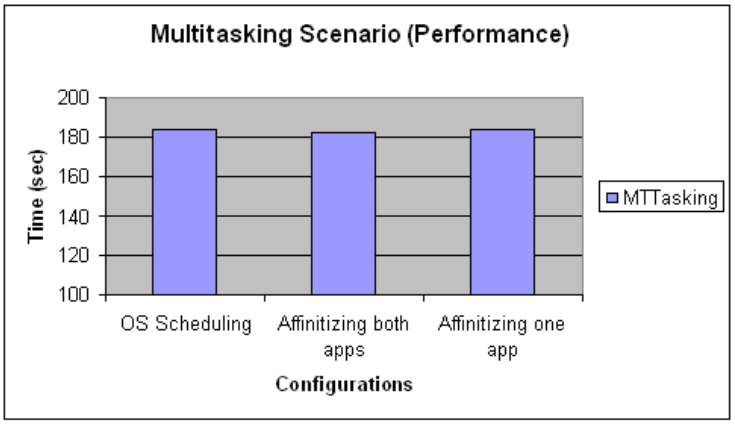


Figure 6: Multitasking Scenario Performance

As shown in Figure 7, CPU power consumption data—the second scenario that is affinitizing both applications to individual cores—demonstrates slightly higher power consumption as compared to Windows XP scheduling.

A similar impact is seen (Figure 8) on platform power consumption, where hard-affinitizing both the applications to each core shows slightly higher platform power consumption as compared to letting Windows XP do the scheduling.

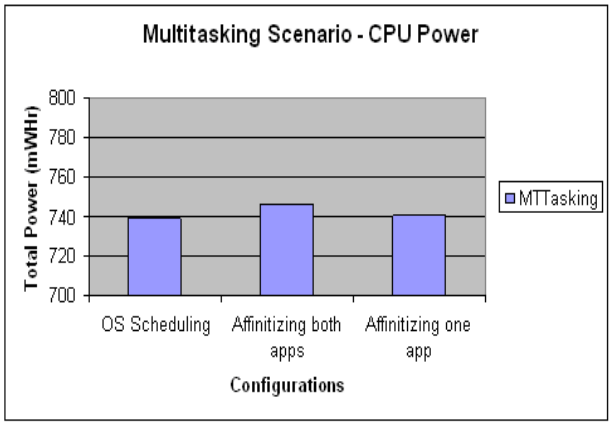


Figure 7: Multitasking Scenario - CPU Power

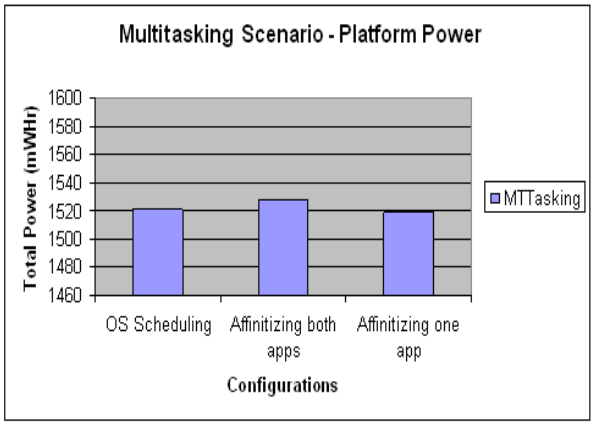


Figure 8: Multitasking Scenario - Platform Power

Summary

The following can be deduced from the threading tests described in this section:

- ***Threading done right provides performance boosts as well as power savings*** - As indicated by the data in the Balanced Threading Model section, a properly multithreaded application demonstrates performance improvement as well as power savings. This includes multithreaded

implementations with a minimum of imbalance and synchronization points. While multithreading an application, we recommend using a threading model in which all the threads perform an equal amount of work independently. Minimizing synchronization points between threads leads to more time spent in the parallel section, which translates to good performance improvements and power savings.

- ***Thread imbalance may cause performance degradation and may not provide power/performance benefits as compared to balanced threading*** - As discussed in the Imbalanced Threading Model section, applications with an imbalanced threading model may show less performance improvement as compared to a balanced threading model, and thus consume more power as compared to the balanced implementation. Thread imbalances may cause frequent thread migration across cores, which may result in reporting incorrect processor activity states. This may lead processors to go down to lower frequency states (if the hotfix provided by Microsoft is not enabled) in adaptive schemes even if one of the threads is utilizing full-processor resources. This issue may occur while running single-threaded applications on a dual-core system in Adaptive mode as well.

- ***Utilize the GV3 hotfix (KB896256) from Microsoft*** - If a multithreaded application indicates performance degradation/increased power consumption in Adaptive (PL) mode, install GV3 hotfix from Microsoft; the issue might be that the OS is getting incorrect information about processor performance while in Adaptive mode. GV3 hotfix will track CPU utilization across the entire package rather than individual cores, and will enable the OS to operate at optimum frequency.

- ***Use OS scheduling vs. hard affinitizing*** - In general, for Intel Core Duo systems, it is advisable to use OS scheduling as opposed to affinitizing threads or applications. The OS scheduler will utilize any underutilized core, while hard affinitizing may potentially degrade performance since an application may need to wait for the availability of a specific processor even though other processors in the system are idle.

Compilers, Performance Libraries, and Instruction Sets

Another way to achieve better computational efficiency and performance is to use an optimizing compiler, performance libraries, and/or make use of advanced instruction sets.

Many compilers now offer OpenMP directives that look for opportunities for parallelism. OpenMP* is a portable, scalable model for developing parallel applications. For example, the Intel® Professional Edition Compilers offer support for creating multi-threaded applications. Features include advanced optimization, multi-threading, and processor support that for processor dispatch, vectorization, auto-parallelization, OpenMP*, data prefetching, and loop unrolling, along with highly optimized libraries. Microsoft Visual Studio 2005 also includes the OpenMP portable threading library.

Performance libraries such as Intel® Performance Primitives (IPP) contain highly optimized algorithms and code for common functions such as video/audio encode/decode, cryptography, speech encoding and recognition, computer vision, signal processing, etc. The Intel® Math Kernel libraries provide highly efficient algorithms for many math and scientific functions including fast Fourier transforms, linear algebra, and vector math.

Advanced instruction sets help the developer take advantage of new processor features that are specifically tailored to certain applications. For example, the Intel Streaming SIMD Extensions (SSE) 4.1 is a set of new instructions tailored for applications that involve graphics, video encoding and processing, 3D imaging, gaming, web servers, and application servers. Optimizing with these instructions will deliver increased performance and energy efficiency to a broad range of 32-bit and 64-bit applications. For more detail on SSE 4.1 and Application Targeted Accelerators, see [14].

Data Efficiency

As stated in the introduction, data efficiency reduces energy costs by minimizing data movement. As summarized in [2], data efficiency can be achieved by designing:

- software algorithms that minimize data movement
- memory hierarchies that keep data close to processing elements
- application software that efficiently uses cache memories

Just as we demonstrated with computational efficiency, data efficiency delivers performance benefits and saves energy. The following sections examine several areas where data efficiency methods can be applied to save energy: DVD playback, native command queuing, and use of cache.

DVD Playback[5]

This section analyzes the power consumption of three different DVD playback software applications, and provides recommendations for reducing power consumption. For this study, three DVD playback applications were analyzed (DVD App #1, #2, and #3) and run with the multiple configurations available while power measurements were taken, primarily maximum power-saving mode vs. no power-saving mode. The workload used for the analysis was a standard definition of DVD content shipped with the MobileMark* 2005[6] benchmarking tool. We provide our recommendation on an optimal strategy for reducing power consumption while playing content from a DVD-ROM. See Appendix A for a description of the test environment used for these experiments.

Test Observations

As a baseline, the measured data in Figure 9 was used to target the energy-savings analysis. Figure 9 indicates that DVD drive spin-up is the most power hungry (requiring over 4W for a brief period) and that reducing spin-up can lead to energy savings. Continuous DVD read consumes about 2.5W of power and is another area for potential savings.

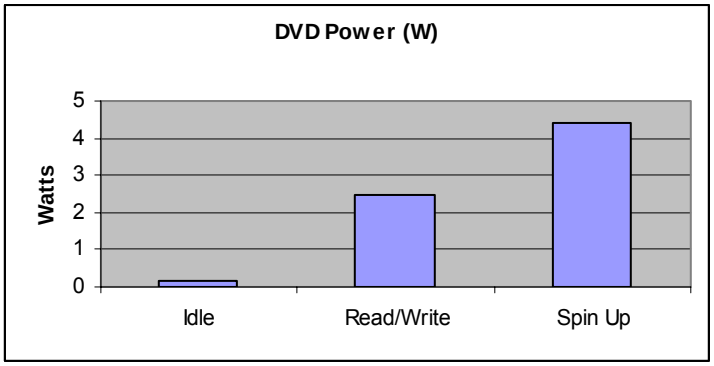


Figure 9: Baseline Power Consumption during Typical DVD Playback States

Figure 10 shows the actual ~20 minute sample from one of the DVD playback applications and is representative of the power demand for all applications operating in maximum performance mode (no power savings). Note that the average power consumed by the DVD player for continuous read is about 2.5 watts.

[5] More detail on this study can be obtained from: http://softwarecommunity.intel.com/articles/eng/1089.htm
[6] For details on MobileMark 2005, see: http://www.bapco.com/

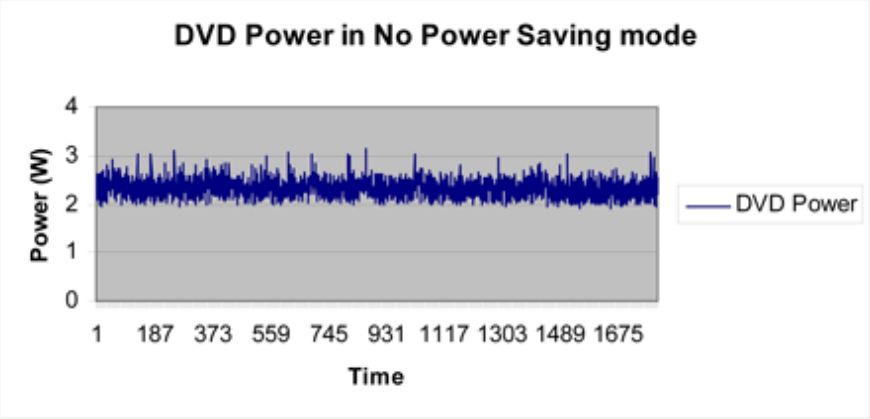


Figure 10: DVD Power from continuous Read

The chart in Figure 11 illustrates the power signature when the DVD mechanism is spun down and no reads are occurring. This signature was evident in Application #1 when it used max power saving mode.

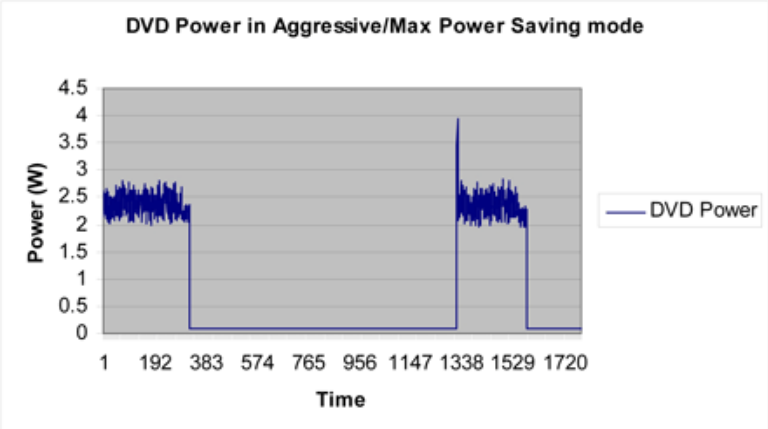


Figure 11: DVD Power consumed in Max Power Saving Mode

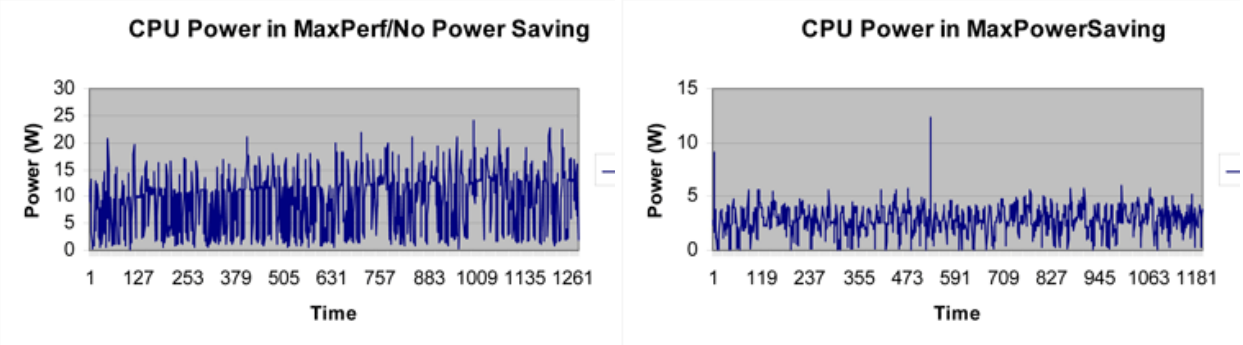


Figure 12: Application #2 CPU Power Profiles

The data in Figure 12 was captured from Application #2, which seemed to exhibit a dramatic difference in CPU power consumption between the no power saving mode and the maximum power saving mode. The reason for this is that in the no power saving mode, the application changed the system power scheme to run with the maximum frequency available and then restored the original power scheme after the run was completed. It's clear from this observation that potential energy savings are possible by considering CPU energy consumption in the software design.

Test Results

Table 1 shows the actual measured energy usage for the three DVD Playback applications using the MobileMark 2005 DVD Playback load. Figure 13 shows the plot of this data. It's valuable to note from this data that difference between the worst case energy consumption (App 2, 10143 mWHrs) and the best case (App 3, 6023 mWHrs) energy consumption is over 4000 mWHrs and appears to be entirely due to the design choices made by the application developers. This is about a 40% energy savings. Even a 10% energy savings on a 4-hour battery would provide 24 minute more battery time.

Application	Mode	DVD Energy (mWHrs)	CPU Energy (mWHrs)	Platform Energy (mWHrs)
DVD App 1	No Save	869.84	663.92	6618.76
	Max Save	263.41	762.99	6039.41
DVD App 2	No Save	897.82	3329.53	10143.56
	Max Save	895.57	1064.02	7509.18
DVD App 3	No Save	780.93	703.25	6202.04
	Max Save	781.04	554.87	6023.57

Table 1: Energy Consumed during DVD Playback

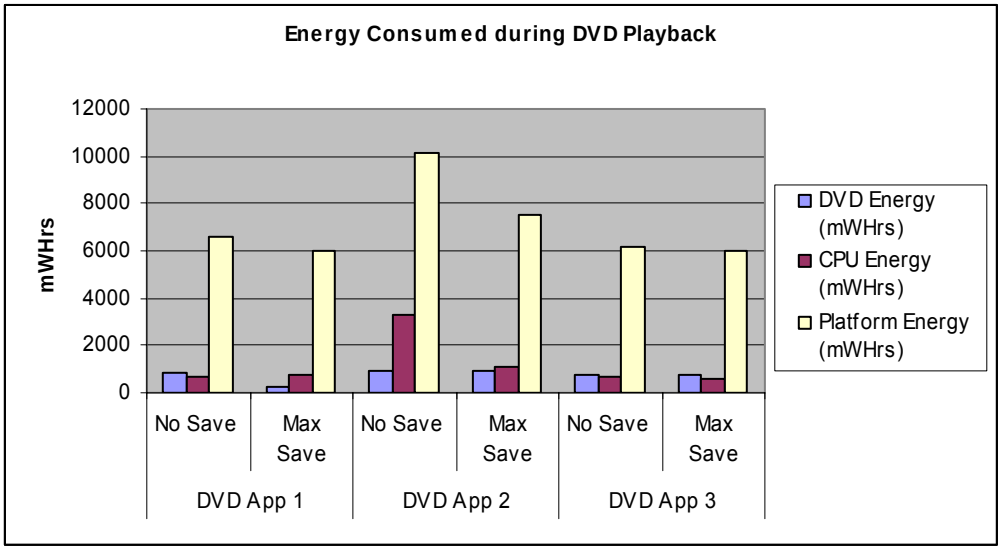


Figure 13: Energy Consumed during DVD Playback

The analysis in Table 2 indicates that the combined energy savings measured for the CPU and the DVD are largely responsible for the overall platform energy savings during playback (from 83 to 87%).

Application	Mode	DVD + CPU Energy (mWHrs)	Platform Energy (mWHrs)	DVD/CPU Save %
DVD App 1	No Save	1533.76	6618.76	87.57%
	Max Save	1026.4	6039.41	
DVD App 2	No Save	4227.35	10143.56	86.08%
	Max Save	1959.59	7509.18	
DVD App 3	No Save	1484.18	6202.04	83.08%
	Max Save	1335.91	6023.57	

Table 2: % Energy Savings Attributed to DVD & CPU

Recommendations

From the results of the studies performed, three guidelines emerge that can help save energy during DVD playback:

- **Buffering**: The studies shown above indicate that the technique of buffering implemented by DVD Playback Application #1 reduces DVD power consumption by 70% and overall platform power consumption by about 10%, as compared to other techniques.
- **Minimize DVD drive use**: It is always recommended to reduce DVD spin-up, spin-downs, and read accesses in order to save power.
- **Let the OS manage the CPU frequency**: We do not recommend changing the CPU power scheme to run the processor at the highest available frequency. The Operating System will apply Intel SpeedStep Technology and automatically change the operating frequency as processing demand increases and bump up the frequency as needed.

Disk I/O[7]

This section summarizes the analysis of power characteristics of the disk during sequential/random reads and native command queuing, and provides an analysis on file fragmentation and disk thrashing. This section also provides guidelines on optimizing the power during disk I/O in various usage models along with the power impact. For additional details as well as sample code sample code, see [3].

Background

The analyses are based on the typical performance characteristics of hard disk drives (HDD) which is affected by RPM, seek time, rotational latency, and the sustainable transfer rate. Furthermore, the actual throughput of the system will also depend on the physical location of the data on the drive. Since the angular velocity of the disk is constant, more data can be read from the outermost perimeter of the disk than the inner perimeter in a single rotation.

When a read request is placed by an application, the disk may have to be spun-up first, the read/write head must be positioned at the appropriate sector, data is then read and optionally placed in OS file system cache, and then copied to the application buffer.

Table 3 shows the relative time (in milliseconds) involved in these operations, based on the theoretical specification of the SATA drive used. While the actual numbers will vary between different drives, this gives a relative idea on the times taken. (Platform specs: Intel® Core™ Duo 2.0GHz, Jamison Canyon* CRB, 2x512MB DDR2, 40GB SATA 5400RPM-2.5'' mobile, Windows* XP-SP2). Figure 14 shows the average power consumed during idle, read/write, and spin-up.

Avg Seek Time (ms)	Avg Latency Time (ms)	Spindle Start Time (ms)
12	5.56	4000

Table 3: HDD Performance Data

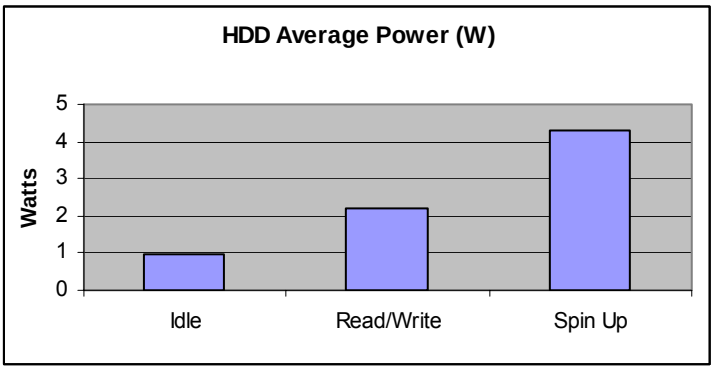


Figure 14: HDD Average Power Consumed

[7] More details from this analysis are available at: http://softwarecommunity.intel.com/articles/eng/1091.htm

It's clear from the data that disk spin-up takes the most time and consume the most power. In fact, the power profile is similar to that of the DVD player described in the previous section. Applications performing disk I/O should take this power profile into consideration and optimize for the power and performance of the hard disk.

Test Results

Five separate experiments were conducted to assess better understand the energy usage of hard disk drives using various I/O methodologies:

- Impact of block size on sequential reads
- Effect of buffering during multimedia playback
- Impact of file fragmentation
- Impact of native command queuing on random reads
- Disk I/O in multi-threaded code

The setup, results, and recommendations are described below. For complete details, see [reference].

Impact of Block Size on Sequential Reads		
Hypothesis		
When reading a large volume of sequential data, reading the data in larger chunks requires lower processor utilization and less energy		
Setup	***Observations***	***Recommendations***
We created a large file (around 1GB) and read the entire file in blocks of various sizes. As a general rule of thumb for any disk I/O, we rebooted the system between runs to avoid any file-system cache interference.	We measured of CPU utilization and energy as we varied block size from 1 bit up to 64KB. As expected, the CPU utilization and the energy required dropped as the block size increased. The CPU utilization and energy usage leveled off with greater block sizes.	• Use block sizes of 8KB or greater for improved performance

Buffering during Multimedia Playback		
Hypothesis		
For multimedia playback, reading ahead and caching media content will save energy		
Setup	***Observations***	***Recommendations***
We compared the energy usage of reading/playing an MP3 file of (4MB) in two ways – reading in ~2KB chunks and reading/ buffering the entire file.	Similar to the DVD playback experiment, when reading the data in small chunks the hard disk remains active and consumes more power than if we read the entire file, access it from buffer, and let the HD go idle.	• Utilize a buffering strategy in multimedia playback to minimize disk reads and save energy

Impact of File Fragmentation		
Hypothesis		
The performance and energy costs to read a fragmented file are greater than that of a contiguous file.		
Setup	***Observations***	***Recommendations***
Store a 256MB file in fragmented and unfragmented states. Read the files and compare the results.	As expected the fragmented file took longer to read – over twice as long – ~26 seconds fragmented and ~11 seconds for the contiguous file. The energy savings were proportional.	• Avoid by pre-allocating large sequential files when they are created, e.g. SetLength in .Net* framework • Use NtFsControlFile() to aid in defragmenting files • End users can defragment their volumes periodically

Impact of Native Command Queuing on Random Reads		
Hypothesis		
Effective use of asynchronous I/O with NCQ improves performance and saves energy.		
Setup	***Observations***	***Recommendations***
We chose a random set of files and started reading at 64KB from a random offset of each file. We compared the use of synchronous I/O to asynchronous with NCQ.	When NCQ was utilized, the total time reduced by ~15% and there was a similar reduction in total energy for the task.	• Applications that deal with random I/O or I/O operations with multiple files should use asynchronous I/O to take advantage of NCQ. • Queue up all the read requests and use events or callbacks to determine if the read requests are complete.

Disk I/O in Multi-threaded Code		
Hypothesis		
The performance and energy costs to read a fragmented file are greater than that of a contiguous file.		
Setup	***Observations***	***Recommendations***
For this analysis, we developed a bitmap-to-JPEG algorithm based on the IJG library that converts a large set of BMPs to JPEGs. We created a serial version of the application and several multi-threaded versions: 1. Two threads that split the files and work independently – competing for disk access 2. Add a thread to coordinate the buffer read/writes and handle requests sequentially 3. Use queued I/O in the coordinating thread to optimize read/writes	Thread solution #1 provided almost no performance improvement over the serial version due to I/O contention and thrashing. Solutions 2 and 3 (buffered I/O and queuing I/O) provided ~1.52x-1.56x scaling) – a significant performance gain. Solution 2 and 3 also yielded ~30% reduction in total energy cost over the serial version.	• For multiple threads competing simultaneously for disk I/O, queue the I/O calls and utilize NCQ. Reordering may help optimize the requests, improve performance, and save energy. • When multiple threads competing for the disk causes significant disk thrashing, consolidate all the read/write operations in a single thread to reduce read/write head thrashing and reduce frequent disk spin-ups as well.

File Transfer over Wireless

Another opportunity to affect data efficiency is by investigating the power consumption of a laptop while transmitting compressed and non-compressed data over a wireless network to determine if there are more power efficient methods. In these experiments we focused on how the compression ratio or size of the file affects power consumption. We did not compare performance of various compression algorithms.

The goal of the research was to obtain data that would help answer questions such as:

- For upload, is it better to compress the data before transmission or leave it uncompressed?
- Is it better to compress a file before downloading?
- How will the wireless adapter, CPU utilization, data compression ratio, and transmission time affect the laptop power consumption?

The general methodology used was to transmit compressed (using GZip 1.2.4) and uncompressed data with varying file sizes and measure platform power with a Fluke NetDAQ[3] . For complete details of the methodology and setup, please see the white paper on this topic.[8] Note that this this power study examined only the client side (laptop) and did not include the server side.

Test Parameters

To achieve reproducible and consistent results, certain parameters were adopted as follows:

1. To minimize noise and interference, a controlled network was used:
    a. A wireless network was set-up in an isolated environment to reduce noise and interference
    b. A dedicated, private network (via access point), with only a single client transmitting data
2. Data sets: Different “txt” and “tif” files with varying compression ratios
3. Compression algorithm:

[8] More detailed coverage of this topic can be found at: http://www.intel.com/cd/ids/developer/asmo-na/eng/333927.htm

a. Gzip 1.2.4* selected since it is open source and easy to customize.
b. Different compression algorithms can affect the compression ratio but the study only focused on the size of compression ratio rather than the algorithm. The differences between Gzip 1.2.4* and other compression algorithms beyond the scope of this study.

4. Test runs were scaled to maintain workloads long enough (in duration) to minimize errors in platform power measurements. Power consumption per run is the average value of 100 iterations.

Data Sets and Test Procedure

The compression ratio of a given data set plays a significant role in determining whether to send/receive uncompressed data or to use compression before transmitting the data. Five different data sets were used, with the corresponding data size and compression ratios shown in Table 4. The compression ratios range from low (1.2x) to high (14.04x).

Data Set	Original size (KB)	Compression Rate Gzip 1.2.4	Description
Tulips.tif	1179	1.2x	Med size file, very low compression ratio
Book1	751	2.45x	Med size file, low compression ratio
World95.txt	2935	5.06x	Large size file, high compression ratio
Pic	502	8.96x	Small size file, high compression ratio
Frymire.tif	3708	14.04x	Large size file, very high compression ratio

Table 4: Data sets were from Jeff Gilchrist Archive Compression Test (ACT)* which are set of benchmarks for data compression. http://www.compression.ca/act/act-files.html*

To perform the tests, the test system (Intel® Core Duo/Customer Reference Board) was network mapped (via access point) to the server's file system and the Windows™ XP internal "COPY" command was used to transfer the data from the client to the server and vice versa over the wireless network. To reduce the affect of anomalous events, each experiment was repeated 100 times while measuring the power. The final power number was divided 100 to determine the average energy used for that data set.

Each of the data sets was transmitted as follows:

- Upload the uncompressed file to the server
- Compress the file and upload the compressed file to the server
- Download the uncompressed file from the server
- Download the compressed file from the server and then uncompress on the client

Wireless Adapter Power Profile

Before looking into various case studies, let us look at the wireless adapter power profile. Table 5 lists the test platform's power profile when wireless adapter is disabled, on with no connection, on and connected to an access point, and on but searching for signal.

Scenario	Average Platform Power (W)	Average WLAN Power (W)	Total
WLAN Radio Off	13.2	0	13.2
WLAN Radio On (no connection to AP)	14.1	0.35	14.45
WLAN Radio On (connected to AP)	14.2	0.45	14.65
WLAN Radio On (searching for AP)	15.7	1.6	17.3

Table 5: WLAN Adapter Average Power Consumption

The wireless adapter uses most power when actively seeking an access point (AP) although this is typically just a brief period of time. When the "radio is on" and the system is connected to the network but not transmitting any data, the average power consumption is ~450mW. While, when searching for AP, the power consumption is ~ 1600mW.

Observations

CPU Profile

We observed that the CPU utilization is high when compressing and uncompressing the data (99-100% when compressing and 84-100% when uncompressing). It drops to 4-7% when transmitting the data regardless if it is compressed or not. As expected the processor frequency goes to maximum (highest Performance Frequency State) when compressing and uncompressing. For transmitting the data over network the processor remains at a lower Performance Frequency state since the CPU utilization is low (4-7%)

Upload Power Consumption

In various runs, the total power consumption of uploading uncompressed data is compared with compressing and uploading the data. Figure 15 shows power consumption comparison for uploading uncompressed data vs. compressing and uploading the data. The secondary Y axis in Figure 15 plots corresponding compression ratio for the given data set. Note that the data sets with higher compression ratios (higher than 1.2x) show benefit for power consumption when compressing first and then uploading the data set. For the data set with the lowest compression ratio (1.2x in this case), uploading uncompressed data is more power efficient by a small amount.

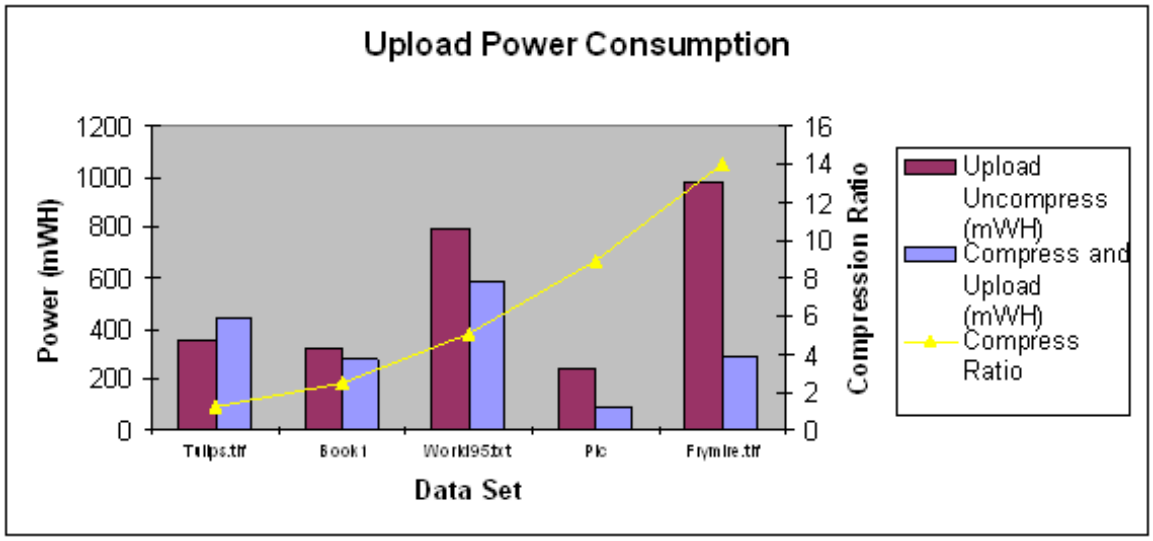


Figure 15: Upload over WLAN Total Power Consumption (Energy)

Download Power Consumption

Similarly, the same data set is used for investigating the download power consumption. Figure 16 indicates the power consumption for downloading uncompressed data vs. downloading compressed data and then uncompressing it.

The graph in Figure 16 indicates average power consumption for each data set as well as the compression ratio on secondary Y axis. As indicated, for data sets with higher compression ratios, downloading compressed data and then uncompressing is more power efficient than downloading uncompressed data. For the data set with the lowest compression ratio (1.2x in this case), downloading uncompressed data is more power efficient. For the 'Book1' data set (compression ratio 2.45x) the power consumption of downloading uncompressed data vs. downloading compressed data and then uncompressing demonstrates minimal difference.

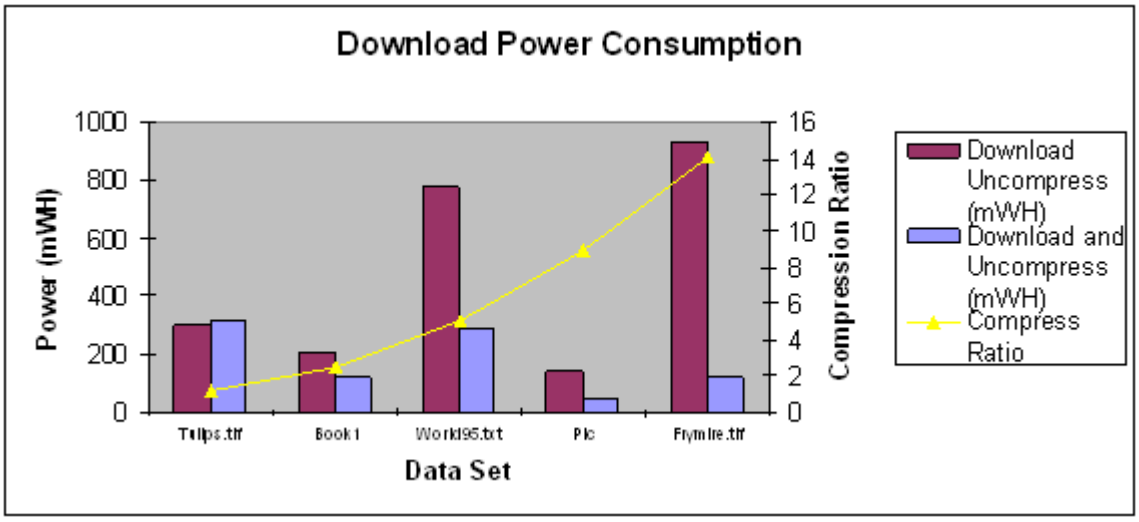


Figure 16: Download over WLAN Total Power Consumption (Energy)

Conclusion

The size of a data file being transferred over a wireless network directly affects the elapsed time to transfer and therefore the power consumption used not only for the wireless adapter but the entire platform. For improved data efficiency and energy efficiency, we recommend the following:

- For data sets with higher compression ratios (more than 3.0x), uploading/downloading compressed data provides better power savings as compared to transmitting uncompressed data. It is beneficial for applications to transmit compressed data for data sets having higher compression ratios.
- For data sets with lower compression ratios (~1.2x in this case which is hardly compressed), compressing the data before uploading/ decompressing after download adds extra overhead. We recommend uploading/downloading uncompressed data in these cases.
- For data sets with compression ratio around 2.5-3.0x, there is a minimal difference in the power saving when uploading/downloading compressed data vs. uncompressed data.

Context Awareness

To save energy through software, we have described techniques for computational efficiency and for data efficiency. A third paradigm to address is context awareness. Context awareness was first introduced by Schilit in 1994 [18]. The objective is to create applications that can respond or adapt to changes in the environment. For the physical environment this requires sensors and the ability to generate events or state changes to which the applications can react. For example, many notebook computers respond to the change from AC to battery power by automatically dimming the display. Some notebook PCs park the hard drive heads when sensors detect that the device is falling – to avoid a head crash. Some applications may write cached data to flash when the battery is getting critically low.

Context may also apply to a user's situation. For example, a software application acting as an intelligent travel assistant may find and offer alternative flights when it's apparent based on your location (stuck in traffic) that you have missed your scheduled departure.

With respect to power, applications should respond to system changes and take actions that will conserve energy – or at least offer options to the user. These events include AC vs. battery, charge remaining on the battery, and the state of various energy consuming devices such as WiFi and Bluetooth radios.

The next section describes techniques for creating more energy efficient games and a later section describes various tools that support the development of context aware applications.

Enabling Games for Power

PC gaming has traditionally been associated with desktop systems due to the performance requirements of game applications. As laptop capabilities become comparable to desktops, some gamers consider these as gaming platforms. Battery life is an important factor for system usability. To determine the most power efficient configuration, this section analyzes the power consumption of laptops while changing different game applications settings like graphics, resolution, FPS, and more. The study focuses on how the different game settings can provide a more efficient power configuration with a battery-powered laptop and provides recommendations on how to optimize games for power.[9]

All the tests described here were conducted on one of two configurations:

1. A 2.0 GHz dual-core Intel Dual Core engineering system with an nVidia 7800 graphics card. Power measurements taken with a Fluke NetDAQ® - see Appendix A
2. A Sony* VAIO, Centrino Core Duo 2.0GHz, with an nVidia 7400 graphics card. Power measurements taken with an Extech Power Analyzer[10]

Three off-the-shelf games were used during the measurements (2 first-person shooter and 1 real-time strategy). The first person shooter games typically run at a high frame rate; while real time strategy games don't scale up the frame rate.

Test Observations and Results

LCD Brightness

A baseline observation that is helpful to understand is based on LCD brightness. Figure 17 shows that a Sony laptop has ~5W (32%) power reduction when using low brightness (0 of 8 bars) as compared to high brightness (8 of 8 bars). The data indicates that reducing screen brightness can help save power. The power saving percentage may change when the system is busy running workloads.

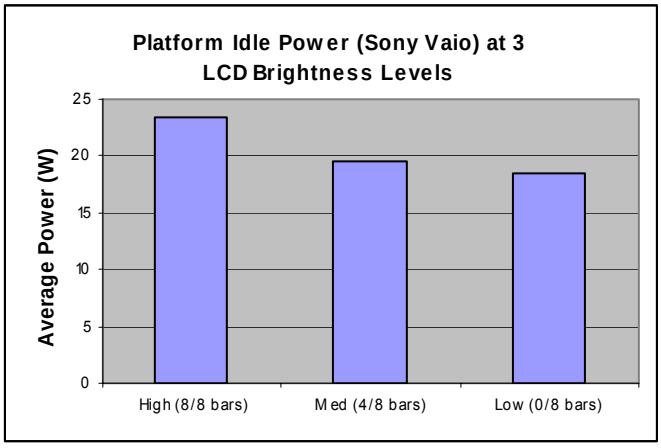


Figure 17: Idle system LCD brightness

Game Resolution Settings

The chosen games support various resolution based on system configuration and game design. Figure 18 shows the effect of power consumption on the Sony laptop with an nVidia 7400 graphics card (Setup #2). The data here represents power scaling, hence lower is better. For example, the blue bar (for game A and B) show baseline/default power consumption data scaled to 1.0. The other bars are scaled accordingly. For game A, with a resolution of 800x600, the power consumption is ~5% less as compared

[9] For complete details of this study, please see: http://www.intel.com/cd/ids/developer/asmo-na/eng/dc/mobile/333574.htm

[10] For details on Extech Power Analyzers, see: http://www.extech.com/instrument/products/310_399/380803Power.html

to baseline; when using the 640x480 resolution, the power consumption is ~8% less as compared to baseline. Similar results were observed on the Engineering system (Setup #1.)

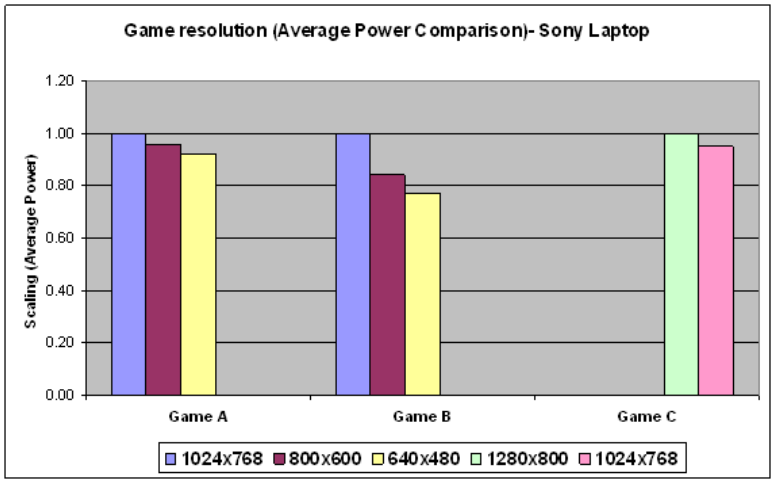


Figure 18: Game resolution and average power comparison on a Sony laptop

Game Quality Settings

The chosen games include options to adjust game quality settings, including 'Textures', 'Shaders', 'Shadow Effect', 'Smoke Effect', 'Water Effect' etc. For example, 'Shaders' provide 3 quality settings: high, medium, and low. Shaders usually determine final surface properties of 3D object. It can include details on lighting effect, shadows, reflections, etc and how these properties affect the object during play. The details on how much pre-processing is needed (finer level details) can be scaled by selecting high, medium, and low quality shaders.

Figure 19 shows the power consumption impact of changing the game quality settings on Setup #1. The blue bar represents baseline/default power consumption data which is scaled to 1.0. The baseline quality settings change with system configuration and graphics card. The 2nd bar represents measurements taken with all quality settings set to low. As indicated in the graph, setting quality options to low saves power across all 3 games. Similar results were obtained using Setup #2.

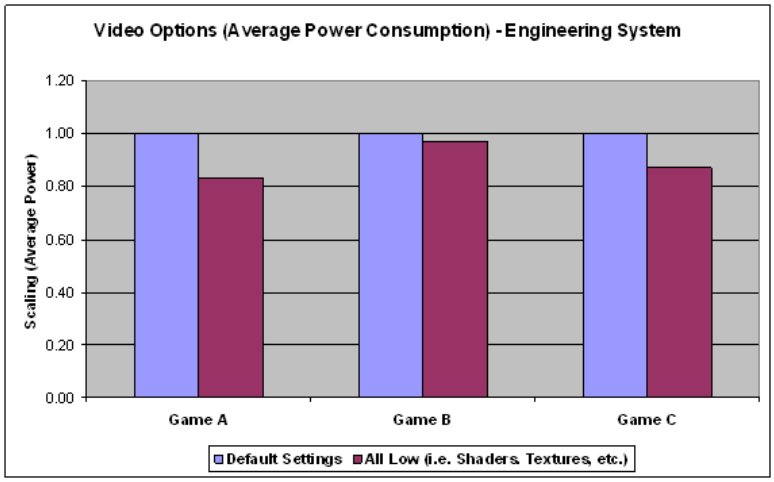


Figure 19: Video options and average power consumption on an engineering system

Game Frame Rate Settings

Another option available in the chosen games is an adjustable frame rate. Reducing the frame rate should reduce the necessary processing and thus save on power. Figure 20 shows measurements taken

on Setup #2 with varying frame rates. Game A runs at 20 frames per seconds as baseline (blue bar), the maroon bar represents 15 FPS and beige bar presents running at 10FPS. Similarly for game B, the baseline is 60 FPS (blue bar), maroon bar represents 30FPS and beige bar represents running at 20 FPS. As indicated in Figure 20, the platform power consumption is reduced when capping the frame rate at lower values. Similar results were observed on the Engineering system (Setup #1.)

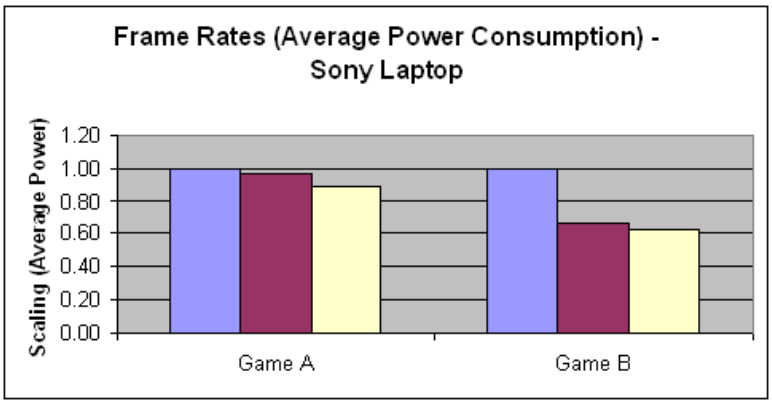


Figure 20: Frame rate and average power consumption on a Sony laptop

Recommendations

It's clear from the experiments that various game and platform settings can influence the amount of power consumed and when set appropriately, can extend battery time. When running on battery power, the following actions can extend play time:

- Reduce display brightness
- Reduce game resolution to lower value
- Reduce the frame rate by capping it to lower value
- Reduce game quality options to lower values

In addition, if WiFi and/or Bluetooth radios are not needed, power can be saved by turning them off. While these recommendations apply to end users, the game options must first be implemented by the game developers. This could be as simple as providing a single choice – "Optimized for Performance" or "Optimized for Power."

Games can be even more user-friendly when they know when to offer choices to the gamer – such as when the user switches from AC to Battery. See the tools section for information on context awareness tools that help developers create applications that can understand the current system state and alter behavior in response to system events.

Operating Systems

This brief section provides some links to references that describe operating system features that support development of power-aware applications – for Microsoft Windows Vista and for Linux.

For creating power-aware applications in Vista, Microsoft has some extensive resources on this topic that can be found at [10,11,12,and 13]. For example, in the whitepaper *Application Power Management Best Practices for Windows Vista* [11], the authors emphasize the following topics:

- Handling Sleep and Resume Transitions
- Preventing System Idle Timeouts
- Designing for Extended Battery Life
- Responding to Common Power Events
- Designing for Entertainment and Media PC Scenarios

- Using Power Management APIs from Managed Code
- Designing for Earlier Versions of Windows
- Responding to Power Events within a Windows Service
- Testing Applications for Power Management

Microsoft also provides information on the Advanced Configuration and Power Interface (ACPI) which defines common interfaces for hardware recognition, motherboard and device configuration, and power management. Writing drivers that are ACPI-compatible allows the OS to control the device power states, power down the device when not in use, and therefore save energy. For more information, see [12]. Most Linux distributions also support ACPI.

For creating power-aware applications for Linux there are a number of groups that can be found simply by searching for "Linux Power Management" via your favored search engine. For example, see [19] for a Linux Power Management Guide or [20] for an article on Power Management in Linux-based systems.

Another consideration for Linux developers is to use an energy-efficient Linux distribution intended for mobile devices such as Midori, an Open Source project for delivering system software on small devices. [21]

An interesting Linux-based tool to check for process energy-efficiency is Linux PowerTOP.[11] PowerTOP identifies processes that, by virtue of tight time interrupts, prevent the CPU from entering deeper C-states. PowerTOP.org has been successful in identifying and improving portions of the Linux OS that have led to energy-efficient improvements in the various distributions.

Tools and Technologies

This section describes a set of tools from Intel that support multi-threading and context-aware computing.

Threading Tools

Intel® Thread Checker

Intel Thread Checker is an analysis tool that pinpoints hard-to-find threading errors such as race conditions and deadlocks in 32-bit and 64-bit applications. It can also be integrated into an automated Quality Assurance/test process to ensure code quality. The tool:

- Detects hidden potential errors such as deadlocks and race conditions, mapping them to the source-code line, call stack, and memory reference
- Displays useful warnings for effective threaded application diagnosis, highlighting the most potentially severe errors
- When used with supported Intel® compilers and source instrumentation mode, tracks the error down to the specific variable in your source code
- With comprehensive error detection, mitigates the risk of adding threads and enables hands-on learning about the fundamental principles of threading

Intel® Thread Profiler

Intel Thread Profiler 3.1 for Windows* helps you tune multi-threaded applications faster, for optimal performance on Intel® multi-core processors. The tool helps the developer:

- Understand what threads are doing and how they interact (timeline view)
- Pinpoint the exact location of performance issues in call stacks and source code to aid analysis
- Measure the number of cores that are effectively utilized by the application to determine actual parallel performance

[11] See: http://LinuxPowerTOP.org

Intel® Thread Building Blocks

Intel Threading Building Blocks 1.1 is a C++ runtime library that abstracts the low-level threading details necessary for optimal multi-core performance. It uses common C++ templates and coding style to eliminate tedious threading implementation work. It allows the developer to create applications that are portable across platforms and inherently scalable, i.e. no code maintenance is required as more processor cores become available.

- Includes commonly needed algorithms designed for parallel performance and scalability
- Provides generic templates to tailor the algorithms to the developer's needs
- Supports easy plug-in deployment into applications to deliver scalable software speed-up, optimizing for both available cores and cache locality
- Reduces the work required to produce threaded software in many cases, by means of pre-built parallel constructs

Intel® VTune™ Performance Analyzer

Intel VTune™ Performance Analyzer simplifies application performance tuning with a graphical user interface and no recompiles required. It is compiler and language independent and works with C, C++, Fortran, C#, Java, .NET and others. VTune includes call graph analysis, sampling events, plus an extensive set of tuning events for all the latest Intel® processors. There is version of VTune for Windows and another for Linux.

Context Awareness Tools

Intel has created several tools for developers that provide features that enable them to develop context-aware applications. These include:

- Mobile Platform Software Developer Kit (MPSDK)
- Laptop Gaming Technology Developer Kit (Gaming TDK)
- Web 2.0 Technology Development Kit (Web 2.0 TDK)

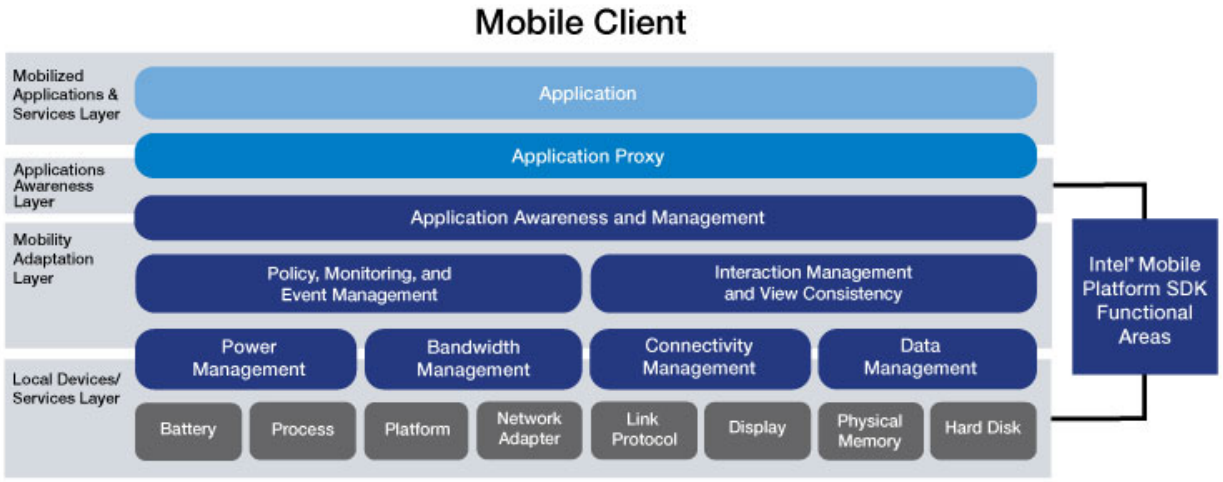
Intel® Mobile Platform SDK

Mobile users expect business applications to adjust automatically to resource-constrained mobile environments. With the Intel Mobile Platform SDK, developers can integrate mobile features that meet user expectations for computing anytime and anywhere:

- **Manage connectivity transparently**: Develop applications that let users focus on their tasks rather than managing network connections
- **Balance power and performance**: Developers can use the power context to get the most from the available battery life
- **Manage available memory**: Use available memory and disk space for local data store and synchronization. Applications developed with the SDK can discover and utilize available memory and disk storage for caching, local data store, and synchronization.
- **Adapt to different display types**: Mobile devices are often connected to different display types. Applications based on the SDK can discover the attached display type and adapt to it.

Creating code that is aware of platform context means that applications can more efficiently adapt to user settings as well as environmental and platform changes.

The following figure shows how the Intel Mobile Platform SDK straddles several different functional layers to provide better integration between applications and mobile hardware.



For more information on the Mobile Platform SDK, see http://ossmpsdk.intel.com

Intel® Laptop Gaming TDK

The Intel® Laptop Gaming Technology Development Kit (TDK) provides an interface to help extend games by adding mobile-aware features to deliver a better laptop gaming experience.

The TDK includes C++ code that help developers create applications that:

- Respond to platform state changes such as power source changes, battery power levels, and WiFi signal strength variations
- Improve game play by monitoring platform characteristics and taking appropriate action in response to changes due to mobile usage
- Provide for better processor utilization in multi-core platforms using threaded API calls

Key features include:

- A top-level API to get information about battery power states, network connectivity, and processor information, including number of cores
- A threaded event management system to monitor system changes with minimal overhead to client game applications.
- Allows creation of user-defined callbacks in an object-oriented way to handle specific platform events

For more information on the Laptop Gaming TDK, visit:
http://softwarecommunity.intel.com/articles/eng/1017.htm

Intel® Web 2.0 TDK

The Web 2.0 TDK provides royalty-free Javascript API, binary, and source code to that enable developers to create web applications that taking advantage of the mobile features on notebooks and UMPCs. In Web 2.0 applications, developers can leverage information about the platform's configuration and context to provide more expressive interactions and better user experience, especially on mobile platforms where intermittent connectivity and limited power are common issues.

The TDK allows developers to learn about the platform's configuration, e.g. display, storage, processor, and the platform's context, e.g. bandwidth, connectivity, power and location, etc. within a browser using JavaScript. It contains documentation and full source code (C++ and JavaScript) for IE 6/7 and Firefox. The code can be incorporated directly in extensions or JavaScript libraries, and be redistributed royalty free.

Key features include:

- Support for Windows XP* with an IE6/7 or Firefox browser
- Creation of user-defined callbacks in an object-oriented way to handle specific platform events
- A high-level JavaScript API to get information about battery power states, network connectivity, and processor information, including number of cores.
- Information APIs for Power, Connectivity, Storage, Bandwidth, Processor, and Location

More information and demos/videos can be found at: http://www.intel.com/software/web20TDK/

Conclusion

This paper examined software methodologies, designs, and software development tools that can be used to improve the energy efficiency of application software and extend mobile platform battery time. Computational efficiency, data efficiency, and context-aware methods can all contribute to creating applications that are power-aware. Intel provides many resources to achieve these goals in the form of white-papers, developer kits, and analysis tools. Many of the resources are referenced above and in the next section. More information on energy-efficiency and other topics can be found at http://softwarecommunity.intel.com

References

1. *Power Analysis of Embedded Software*, Tiwari, Malik, and Wolfe, International Conference on Computer Aided Design, 1994.
2. *Data-Efficient Software and Memory Architectures are Essential for Higher Performance and Lower Power,* Dr. Guido Arnout, Information Quarterly, V.4, No.3, 2005
3. *Power Analysis of Disk I/O Methodologies*, Karthik Krishnan and June De Vega, http://softwarecommunity.intel.com/articles/eng/1091.htm
4. *Power Optimization: Furthering the Mobile Vision*, Karthik Krishnan, Rajshree Chabukswar, and Jun De Vega, http://softwarecommunity.intel.com/articles/eng/1092.htm
5. *Maximizing Performance and Energy-Efficiency on Intel Core Micro architecture using Multi-Threading*, Rajshree Chabukswar, http://www.intel.com/technology/magazine/computing/mobile-power-saving-0506.pdf
6. *Maximizing Power Savings on Mobile Platforms,* Rajshree Chabukswar, http://www.intel.com/cd/ids/developer/asmo-na/eng/274896.htm
7. *Selected Papers on the Analysis of Algorithms,* Donald E. Knuth, 2000
8. *Art of Computer Programming,* 4 Volumes, Donald E. Knuth, 2005
9. *An Introduction to the Analysis of Algorithms,* Robert Sedgewick and Philippe Flajolet, 1996
10. *Windows Vista: Developing Power-Aware Applications*, slide presentation, Pat Stemen and Geralyn Miller, SDC 2005 (see: http://download.microsoft.com/download/c/d/5/cd5154e8-d825-4e14-89c8-8b0eb9dda203/pdc_2005_developingpower-awareapplications.ppt)
11. *Application Power Management Best Practices for Windows Vista*, Microsoft whitepaper, (see: http://www.microsoft.com/whdc/system/pnppwr/powermgmt/PM_apps.mspx)
12. *ACPI / Power Management - Architecture and Driver Support,* Microsoft Hardware Developer Central,(see: http://www.microsoft.com/whdc/system/pnppwr/powermgmt/default.mspx)
13. *Processor Power Management in Windows Vista and Windows Server 2008,* Microsoft whitepaper,(see: http://www.microsoft.com/whdc/system/pnppwr/powermgmt/ProcPowerMgmt.mspx)
14. *Extending the World's Most Popular Processor Architecture,* R.M. Ramanathan et.al., Intel whitepaper, (see: http://download.intel.com/technology/architecture/new-instructions-paper.pdf)
15. *PeekMessage: Optimizing Applications for Extended Battery Life,* Dale Taylor, Intel whitepaper
16. *Energy-Efficient Graphical User Interface Design*, Vallerio, Zhong, Jha
17. *Data Transfer over Wireless LAN Power Consumption Analysis,* Jun De Vega and Rajshree Chabukswar, Intel whitepaper, http://www.intel.com/cd/ids/developer/asmo-na/eng/333927.htm
18. B. Schilit, N. Adams, and R. Want. (1994). "Context-aware computing applications". IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94), Santa Cruz, CA, US: 89-101. (see: http://sandbox.parc.com/want/papers/parctab-wmc-dec94.pdf)
19. *Linux Power Management Guide,* http://www.gentoo.org/doc/en/power-management-guide.xml
20. *Power Management in Linux-based Systems,* http://www.linuxjournal.com/article/6699
21. *Midori,* http://midori.sourceforge.net

About the Authors

The short version of Author's bio, and link to Author's page

Bob Steigerwald

Bob Steigerwald is an engineering manager in Intel's Software Solutions Group at Intel in Folsom, California. He received his B.S. degree in Computer Science from the US Air Force Academy, Masters in CS from the University of Illinois, and Ph.D. in CS from the Naval Postgraduate School where his research was in Software Engineering and software reuse. Currently his team works on defining tools and technologies to support the development of energy-efficient software for Intel-based mobile platforms. His e-mail is bob.steigerwald@intel.com.

Rajshree Chabukswar
Rajshree Chabukswar is a software engineer working on client enabling in the Software Solutions Group that enables client platforms through software optimizations. Prior to working at Intel, she obtained a Masters degree in Computer Engineering from Syracuse University, NY. Her e-mail is rajshree.a.chabukswar@intel.com.

Karthik Krishnan
Karthik Krishnan is a software engineer with the Software Solutions Group at Intel. He holds a Masters degree in Mathematics from the Indian Institute of Technology. His current focus is on power and performance optimization of software applications on dual-core platforms. His e-mail is karthikeyan.krishnan@intel.com.

Jun De Vega
Jun De Vega is an Application Engineer in Intel's Software and Solutions Group, working on application tuning and optimization for Intel® architecture. He supports enabling of ISV applications on Intel® Mobile and Desktop Platforms. Contact him at rodolfo.de.vega@intel.com.

Appendix A - Power Measurement Methodology

Measuring power usage of individual components in a mobile platform is not a trivial task. Various tools exist to provide a high-level estimate of the power consumed by a particular mobile platform, but they do not provide the granular details on specific components. A more accurate but invasive way to measure power will be to use data acquisition (DAQ) tools where specific hardware components are instrumented and a more granular power measurement can be logged. The following lists the platform details we used for our analyses, along with the power-measurement methodology.

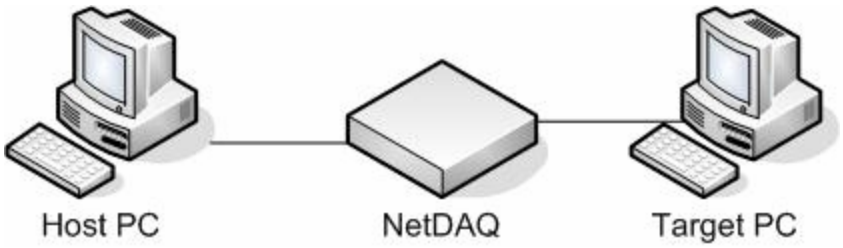
Hardware

Fluke NetDAQ* 2686A
Target PC: Intel® Core™ Duo/2GHz Yonah, Jamison Canyon* CRB, 2x512MB DDR2, 40GB SATA 5400 rpm (2.5” mobile), CD/DVD drive, Microsoft Windows* XP Professional SP2
Host PC: Any IA32 system

Software

Test Applications (different applications used)
NetDAQ Logger: Fluke DAQ Software v2.2

Test Setup



- The Host PC can be any IA32 system with Microsoft Windows XP and the NetDAQ logger software. The logger collects the measured current and voltages and lets the user calculate the average power (W). The sampling interval we used for our entire analysis was 25 milliseconds. The platform power measurement does not include the LCD display of the mobile device.

- The NetDAQ has modules that are attached (individual wires) to the Target PC and measures the current and voltage drop across the sense resistors. The NetDAQ is connected to the Host PC via a cross-over network cable.

- The Target PC (Napa/Yonah) has a special motherboard (Jamison Canyon CRB) with built-in sensors. For each target component (i.e., the CPU), all sense resistors are wired (soldered) at both ends and connected to a module attached to the NetDAQ unit.